1. איני מכירה CRM, לצורך מימוש הפורטל הקמתי 5 טבלאות SQL ע"פ תרשים ה- ERD הבא:

תמונה שמכילה טקסט, תרשים, צילום מסך, קו

התיאור נוצר באופן אוטומטי

**הסבר התרשים:**

תרשים זה מייצג ישויות ואת הקשרים ביניהם.

**PK- מפתח ראשי. FK- מפתח זר.**

**ישויות:**

* **User-** משתמש שהPK שלו זה Id באפשרותו לבצע את פעולת Login לפורטל (באמצעות שם משתמש וסיסמא).
* **Employee-** מייצג עובד שהPK שלו זה Id ישות זו מחזיקה בכל פרטי העובד.
* **Favorites-** מייצג מועדפים שהPK שלו זה Id ישות זו מחזיקה את כל הנתונים המתארים העדפה. (אופציונלית לכל **Employee** .)
* **Type-** מייצג תיאור של סוג המשמש לתיאור סוג העדפה (יישום, שירות, קישור)

**קשרי גומלין:**

* **User\_Id -** שבטבלת **Employee** הוא FK של טבלת **User** (Id).

**הסבר:** שיוך עובד למשתמש. פעולה זו מאפשרת לי לתאר מיהו המשתמש הרשאי לעדכן נתונים של עובד זה.

* **Type\_Id -** שבטבלת **Favorites** הוא FK של טבלת **Type** (Id).

**הסבר:** שיוך סוג להעדפה. פעולה זו מאפשרת לי לתאר את סוג ההעדפה שנבחרה להעדפה ספציפית מסך כל הסוגים שיש (3 סה"כ).

* טבלת **Employee Favorites-**  טבלה זו מכילה רק נתונים המייצגים את הקשר בין טבלת המועדפים לטבלת העובדים. בצורה זו יש לי אפשרות לעקוב אחר כל העדפות של עובד מסוים. תוך שמירה על נתוני המועדפים שיישארו ללא כפילויות.

**תיאור תהליך:** בהוספת מועדף לעובד תחילה נבדוק אם קיימת העדפה כזו עם הנתונים שהוקשו במידה וכן קיים נוסיף רשומה לטבלת **Employee Favorites** את מזהה ההעדפה עם מזהה העובד. במידה ולא קיים תחילה ניצור את נתוני ההעדפה בטבלת **Favorites** ורק לאחר מכן נתאר את הקשר בין ההעדפה החדשה לעובד באמצעות טבלת **Employee Favorites**. כפי שהוסבר לעיל.

* **הסבר:**

**Employee\_Id–** שבטבלת **Employee Favorites** הוא FK של טבלת **Employee** (Id).

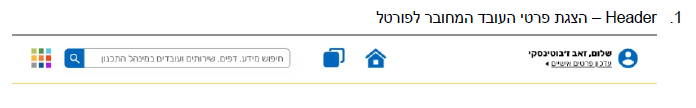
**Favorites\_Id –** שבטבלת **Employee Favorites** הוא FK של טבלת **Favorites** (Id).

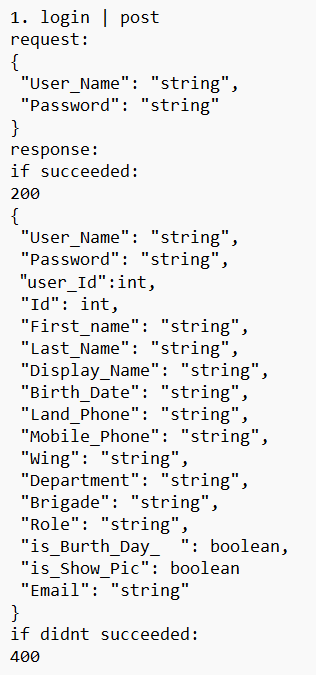
הPK בטבלה זו היא שני השדות שלה המפורטות לעיל. פעולה זו מונעת כפילויות בשיוך העדפה למשתמש.

**הערות:**

* ב- DATABASE אמיתי, מקובל לשמור סיסמאות בצורה מוצפנת.
* העמודות Address ו-Link שטבלה Favorites הן עבור מועדף מסוג לינק (כפי שמופיע בדוגמא) במקרה ונבחר מועדף מסוג יישום/ שירות, ניתן להוסיף לטבלה עמודות רלוונטיות לכל TYPE ולאפשר להכיל NULL בעמודות המיועדות למועדפים האחרים.

2. להלן מבני ה- REQUEST וה- RESPONSE של שירותי בקשות ה- API בהתאמה לארבעת הרכיבים הנדרשים:





תמונה שמכילה טקסט, צילום מסך, פני אדם, אתר

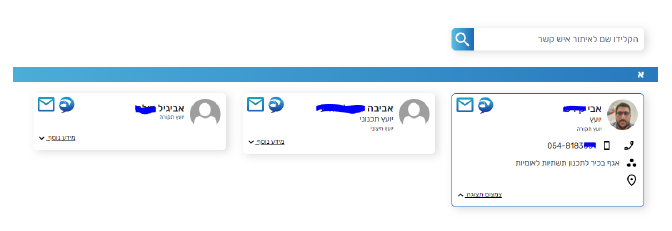
התיאור נוצר באופן אוטומטי

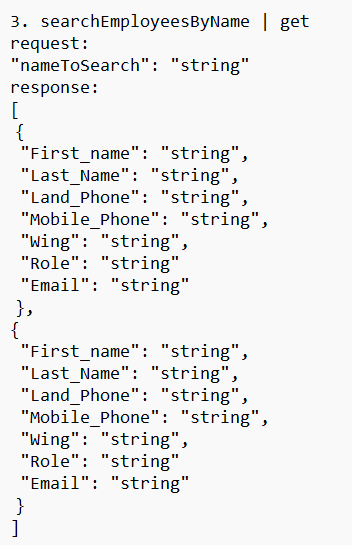
תמונה שמכילה טקסט, גופן, צילום מסך, מסמך

התיאור נוצר באופן אוטומטי

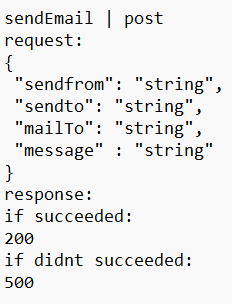
**הערה:**

כמובן שלא תהיה אפשרות לעדכן את כל השדות, לדוגמא מספר הזהות יהיה readonly, תצוגתי בלבד.





החיפוש יחזיר את כל העובדים ששמם הפרטי או שם משפחתם מכיל את התווים שהוזנו (התנאי לסינון יהיה LIKE ולא =)



תמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי

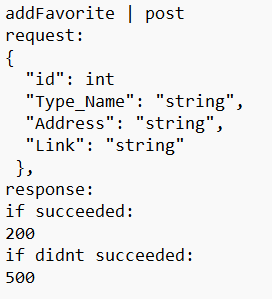
תמונה שמכילה טקסט, גופן, צילום מסך

התיאור נוצר באופן אוטומטי

**הערות:**

ה- ID שנשלח ב- REQUEST הוא המזהה של העובד.

ה- ID שחוזר ב- RESPONSE מכיל את המזהה של המועדף.



3. מצורפים קבצי הקוד.

בחרתי לפתח את ה-API עם המתודות הנדרשות דווקא בשפת שרת NODE.JS מכיוון שכבר פיתחתי באמצעות שפה זו את צד ה- SERVER בפרויקט הגמר שלי בסיום לימודי ההנדסאות והשתמשתי בבסיס נתונים רלציוני ב- SERVER SQL שהיה נראה לי מתאים למימוש מבנה הנתונים בהתאם לפעילות המתוארת של הפורטל.

**הערה כללית:**

במערכות אמת בד"כ בעת בקשת LOGIN חוזר TOKEN המכיל רצף של תווים מוצפנים ובכל קריאת שרת נוספת הוא נשלח ל- SERVER ב- HEADER וכך מתבצע עדכון הנתונים בצורה מאובטחת, כאן בפרויקט הדמה לא ירדתי לרזולציות עכ"כ.